САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И СИСТЕМ

Мозин Виталий Романович

Выпускная квалификационная работа бакалавра

Применение GAN подхода в задаче распознавания образов

Направление 010400

Прикладная математика и информатика

Научный руководитель,
доктор физ.-мат. наук,
профессор
Веремей Е.И.

Санкт-Петербург

2017

Содержание

Введение

В настоящее время технологии распознавания образов достигли серьёзных успехов, в особенности с использованием глубокого обучения [1]. Построение сложных иерархических моделей позволяет решать различные задачи, и значительного прогресса здесь удалось добиться благодаря так называемым дискриминативным (discriminative) моделям. Их основной функцией является определение принадлежности какого-либо объекта к одному из заранее известных классов. Дискриминативный подход интуитивно понятен, здесь можно провести аналогию: человек тоже в своем развитии учится отличать предметы друг от друга, тем самым познавая окружающий мир. Главными представителями данного подхода являются, например, многослойные персептроны, решающие деревья, метод опорных векторов (SVM).

Существует альтернативная концепция машинного обучения, базирующаяся на использовании порождающих (generative) моделей. В отличие от дискриминативных моделей здесь базовый принцип можно сформулировать так: «понять – значит повторить». Действительно, если человек хорошо понимает, что из себя представляет объект, то ему не составит труда детально описать его, он с легкостью сможет самостоятельно «создать» копию этого предмета.

Недавно была предложена [2] «гибридная» концепция Generative Adversarial Networks (GAN), объединяющая дискриминативную и порождающую модели в единую обучаемую систему. Сущностью этого подхода является игра, соревнование порождающей и дискриминативной модели, в ходе которой каждая из них будет обучаться исключительно за счет своего «соперника», и результат может быть достигнут только при должном вкладе обоих участников. Опубликовано множество применений этого подхода для решения различных задач [3,4,5], одна из них будет рассматриваться далее.

Постановка задачи

Одним из важнейших вопросов, связанным с распознаванием образов, является задача повышения разрешения изображений (super-resolution). Известно множество различных подходов к решению этой проблемы [6,7,8,9,10], в данной работе будут рассматриваться два алгоритма, в основе которых лежит GAN подход, целью исследования будет их сравнительный анализ.

Первый из рассматриваемых подходов – SRGAN [7] является специализированным именно для данного типа задач. Второй - Semantic Image Inpainting with Perceptual and Contextual Losses [6] – изначально предложен для проблемы, называющейся дорисовкой (inpainting). Ее основная цель – восстановить изображение, на котором присутствуют некоторые дефекты или шумы. В действительности, эту задачу можно рассматривать как более общую постановку задачи повышения разрешения, поскольку здесь также необходимо каким-то образом восстановить отсутствующие пиксели. Главное отличие в том, что в проблеме дорисовки испорченные пиксели могут быть произвольными, а в задаче увеличения разрешения исходные пиксели заданы на регулярной решетке с определенным шагом.

Обзор литературы

В первую очередь необходимо отметить статью [2], в которой впервые была предложена концепция GAN подхода, и подробно освещены теоретические выкладки, обосновывающие данную схему и алгоритм ее реализации. Впоследствии тем же автором была опубликована работа [3], где собраны воедино различные способы применения данного подхода, рассмотрены его модификации для каждого типа задач. Одним из таких типов является, например, синтез изображения по исходному тексту, описанный в [4].

Также в следующих статьях представлены два алгоритма, рассматриваемые в данной работе: алгоритм Semantic Image Inpainting with Perceptual and Contextual Losses [6] и SRGAN [7]. Для достижения всеобъемлющего обзора методов решения проблемы повышения разрешения изображения можно обратиться к работам [8,9,10], представляющим альтернативные алгоритмы для данной задачи.

Книга [11] содержит обширные теоретические материалы по искусственным нейронным сетям, в том числе дает глубокий анализ многослойного персептрона и методу его обучения.

В интернет-источнике [1] описана концепция и идея сверточных нейронных сетей, разобраны все основные детали, а также приведены примеры решения задач с их помощью. При этом работа [13] содержит предложения о том, как данные сети могут быть использованы при реализации GAN подхода.

В статье [12] представлено предложение о способе нормализации данных для повышения скорости обучения сверточных нейронных сетей. Описанный метод будет применен в данной работе.

Глава 1. GAN подход

1.1. Основная концепция GAN подхода

Generative Adversarial Networks (GAN) является игрой двух нейронных сетей: дискриминативной и порождающей. Интуитивно данный подход можно описать следующим примером: пусть имеется два участника игры – фальшивомонетчик и банкир. Задача первого заключается в том, чтобы научиться создавать собственные банкноты, неотличимые от настоящих. Целью второго игрока, напротив, является определение, какой объект перед ним: поддельный или нет. В процессе соревнования между друг другом, участники улучшают свои основные умения: фальшивомонетчик видит, какие результаты его деятельности проходят проверку, а какие нет, и соответственно корректирует создаваемые им объекты. Банкир, в свою очередь, понимает, по каким признакам можно распознать, настоящая перед ним монета или поддельная, и обучается лучше справляться с поставленной задачей. В идеальной ситуации данный процесс должен привести к тому, что первый игрок будет создавать банкноты никак не отличимые от настоящих. В такой ситуации банкиру ничего не будет оставаться делать, как только гадать. В результате, будет создана порождающая модель, которая совершенным образом понимает, как выглядят деньги.

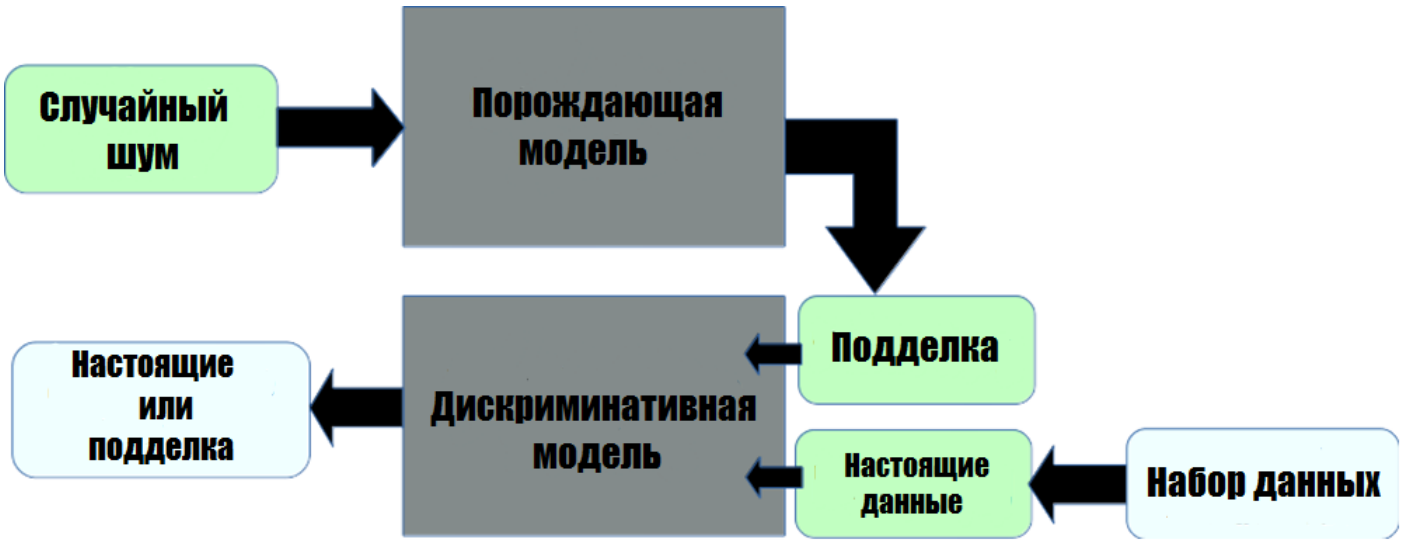

Рис. 1: Схема GAN подхода.

Формальное описание GAN подхода (рис.1) выглядит следующим образом: пусть имеется набор данных **X**, распределенных с помощью некоторого закона $p_{data}(x)$. Обозначим $G(z;\ \theta_g)$ порождающую модель, которая на вход принимает некоторый случайный шум $p_z(z)$, а на выходе выдает распределение p_g. Здесь θ_g - параметры нейронной сети. Введем в рассмотрение дискриминативную модель $D(x;\ \theta_d)$, результатом работы которой является скаляр, отражающий вероятность того, что входной вектор ***x*** поступил из исходного набора **X**, а не из p_g . Основная задача – обучить сеть D максимизировать вероятность правильной идентификации входного объекта к классу: настоящий или подделка. Параллельно сеть G стремится минимизировать $log(1 - D(G(z)))$, т.е. обучиться создавать данные, которые D будет принимать за исходные. Другими словами, описанная схема является минимаксной игрой с целевой функцией $V(G; D)$:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Для обучения GAN модели в статье [2] предлагается использовать следующий алгоритм:

for *количество итераций обучения* **do**

for *k раз* **do**

- *Взять подмножество* $\{z_{(1)}, \dots, z_{(m)}\}$ *из шума* $p_z(z)$
- *Взять подмножество* $\{x_{(1)}, \dots, x_{(m)}\}$ *из исходного распределения* $p_{data}(x)$
- *Изменить параметры дискриминативной модели с помощью стохастического градиента:*

$$\nabla_{\theta_g} = \frac{1}{m}\sum_{i=1}^{m}[logD(x^{(i)}) + \log\left(1 - D\left(G(z^{(i)})\right)\right)]$$

end for

- *Взять подмножество* $\{z_{(1)}, \dots, z_{(m)}\}$ *из шума* $p_z(z)$
- *Изменить параметры порождающей модели с помощью стохастического градиента:*

$$\nabla_{\theta_g} = \frac{1}{m}\sum_{i=1}^{m}[logD(x^{(i)}) + \log\left(1 - D\left(G(z^{(i)})\right)\right)]$$

end for

Основной особенностью представленного алгоритма является тот факт, что на каждый такт обучения порождающей модели приходится несколько итераций обучения дискриминативной. Это делается для того, чтобы во время всего процесса тренировки ни одна из сетей не «обгоняла» другую. Правда, в проводимых при подготовке данной работы экспериментах было замечено, что дискриминативная сеть, напротив, опережает по скорости обучения порождающую, особенно на первых стадиях. Поэтому все последующие результаты приводятся с учетом того, что на каждое обновление параметров D сети следует два обновления G сети (таким образом $k = 2$, оно было выбрано эмпирическим путем).

1.2 Многослойный персептрон

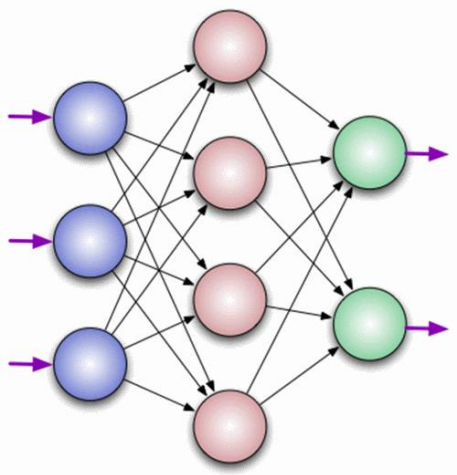

Рис. 2: Слои нейронной сети.

Самым простым кандидатом на роль нейронных сетей в GAN подходе является многослойный персептрон. Любая нейронная сеть состоит из слоев, содержащих заранее заданное количество нейронов. Минимальное количество слоев, из которых может состоять сеть, - два: входной и выходной (на рис.2 отмечены соответственно синим и зеленым цветом). Между ними может находиться любое количество других слоев, называемых скрытыми. (на рис.2 единственный скрытый слой выделен красным цветом). Нейронные сети, которые содержат скрытые слои, называются многослойными.

Каждый слой состоит из нейронов. В детальном рассмотрении нейрон представляет из себя следующее: ему на вход подается вектор сигналов, компоненты которого далее умножаются на коэффициенты, называемые синаптическими весами, и суммируются. После этого результат передается в активационную функцию, выход которой и является итогом работы нейрона.

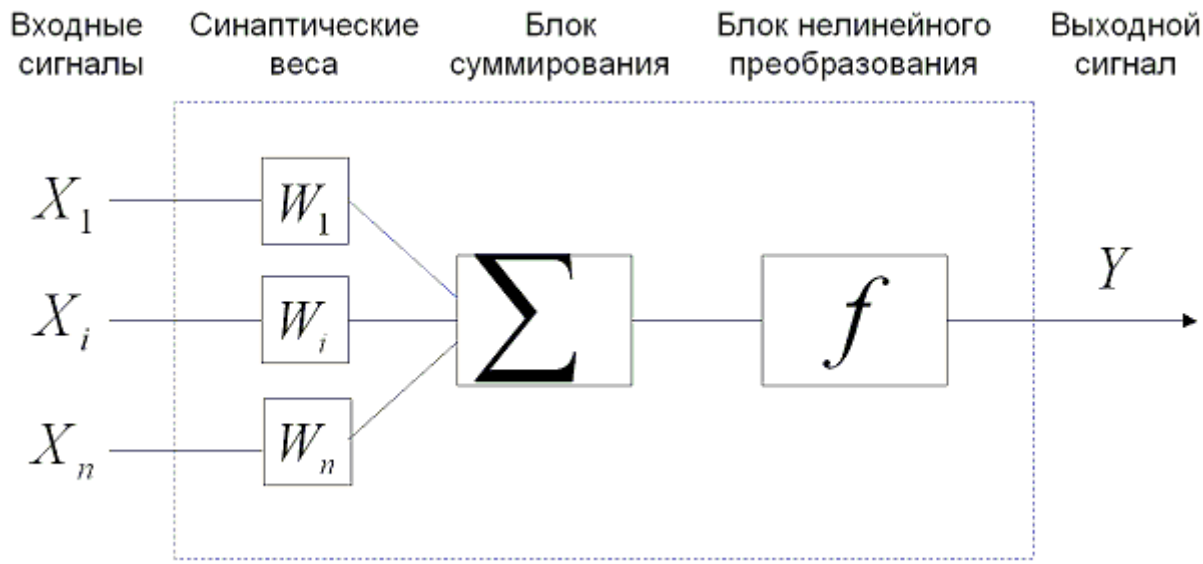

Рис. 3: Схема искусственного нейрона.

Визуальная интерпретация описанной выше схемы работы нейрона представлена на рис.3. С формальной точки зрения работа одного нейрона записывается следующим образом: пусть задано множество входных сигналов $x_1, x_2, \dots, x_n$. Вместе они записываются в виде вектора $\boldsymbol{X}$, принадлежащего n-мерному пространству. После этого каждый сигнал умножается на свой синаптический вес $w_1, w_2, \dots, w_n$, которые также могут быть представлены в виде вектора $\boldsymbol{W}$. В итоге сумма произведений сигнала на соответствующий ему вес будет выглядеть как $\boldsymbol{W}^T\boldsymbol{X}$. Это значение в свою очередь принимается активационной функцией f, результат которой и будет выходным сигналом нейрона. Весь этот процесс формально записывается следующим образом:

$$Y = f(\boldsymbol{W}^T\boldsymbol{X})$$

Стоит отметить, что активационная функция f задается заранее и считается известной. Поэтому из представленной выше формулы нетрудно заметить, что единственным неизвестным параметром является вектор $\boldsymbol{W}$. Основным способом его нахождения, а тем самым обучения нейронной сети, является метод обратного распространения ошибки, детально описанный в [11].

1.3 GAN с многослойным персептроном

Многослойный персептрон является простой структурой для построения GAN модели, отсюда возникает вопрос: насколько полученные результаты будут удовлетворять основной цели – созданию похожих на настоящие объектов. В выбранной при проведении экспериментов архитектуре нейронные сети являются двухслойными, с одинаковыми активационными функциями: ReLU (рис. 4) в первом слое, сигмоидальной во втором (рис. 5).

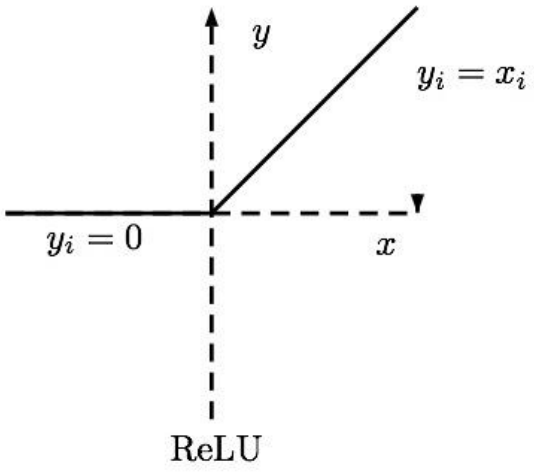

Рис. 4: ReLU.

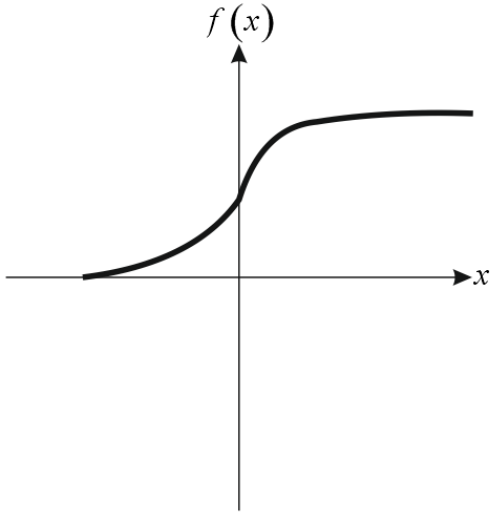

Рис. 5: Сигмоидальная функция.

Результатом обучения на наборе рукописных цифр MNIST являются полученные изображения, представленные на рис.6.

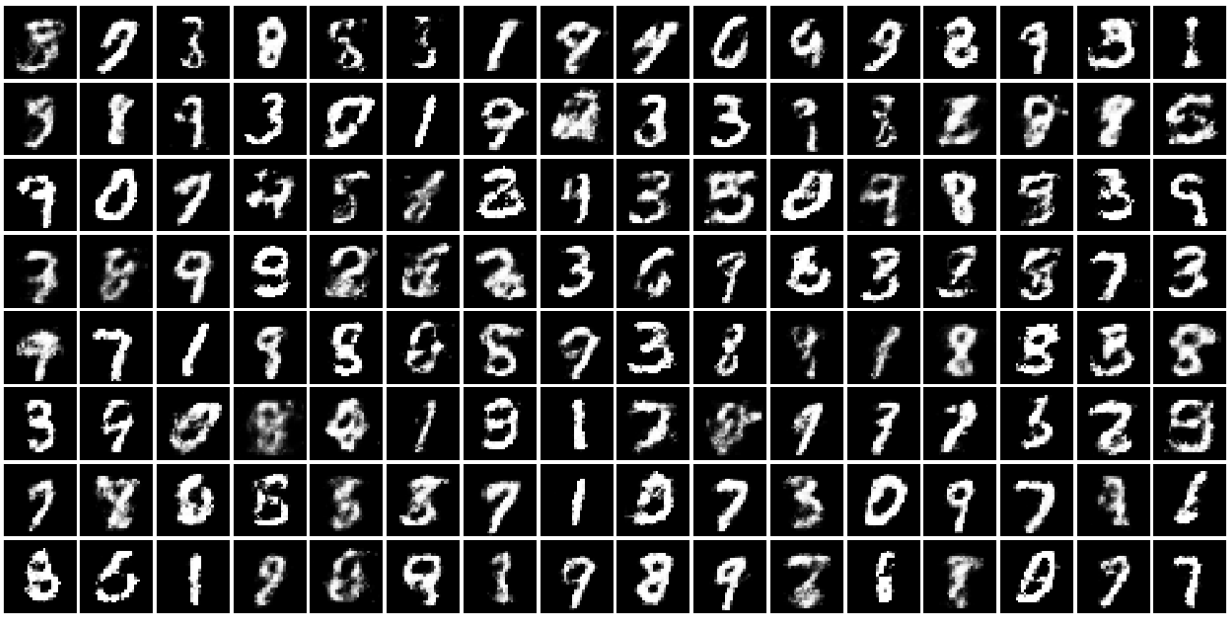

Рис. 6: Результаты GAN подхода с использованием многослойных персептронов.

Необходимо заметить, что наряду с удачными результатами, по которым можно утверждать, что порождающая модель научилась создавать цифры, довольно часто встречаются и более размытые изображения, а некоторые даже трудно идентифицируемые. Данный факт свидетельствует о том, что для высокого качества изображений, что является важным аспектом поставленной задачи, потребуется усложнить структуру нейросетевой модели.

Глава 2. DCGAN подход

2.1 Сверточные нейронные сети

Deep Convolutional Generative Adversarial Networks (DCGAN) по своей сути ничем не отличается от GAN подхода. Дополнение в названии связано с тем, что в качестве дискриминативной и порождающей модели выбираются сверточные (convolutional) нейронные сети (СНС), подробно описанные в [1]. Все остальные принципы сохраняются неизменными.

Сверточные сети являются главными представителями области машинного обучения, называемой глубоким обучением. В своем классическом виде СНС реализуют задачу классификации, т.е. относят входной объект (главным образом – изображение) к одному из заранее заданных классов. Их основной особенностью являются сверточные слои, благодаря которым и достигается вся мощность рассматриваемых сетей.

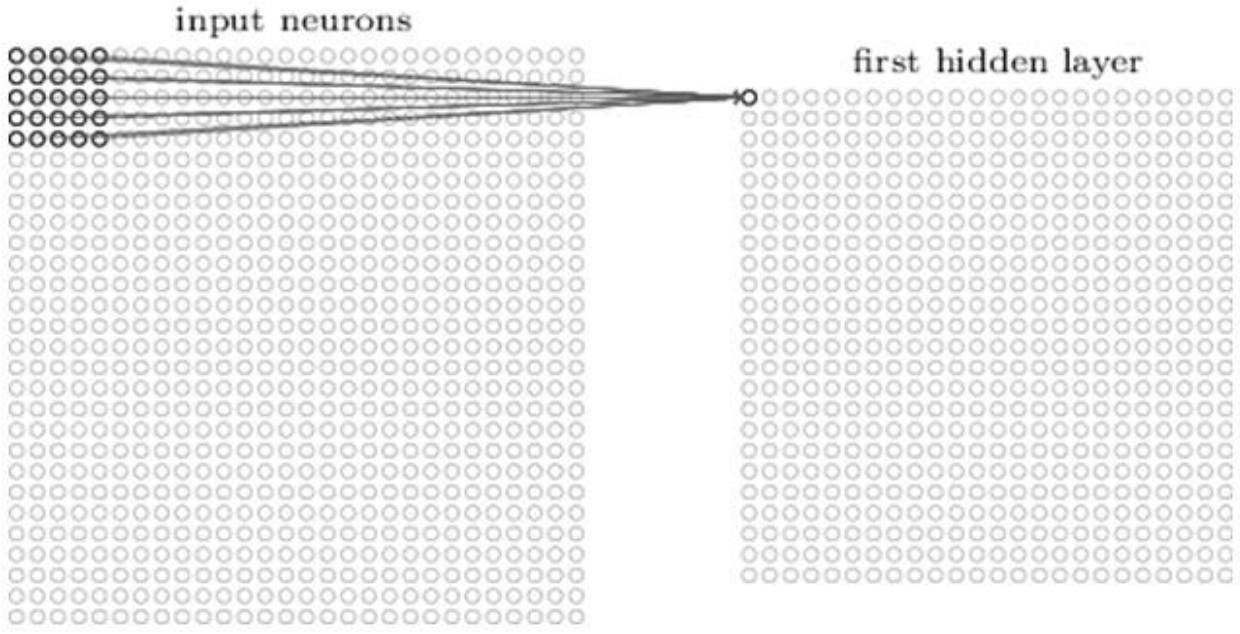

Рис.7: Визуализация фильтра 5x5 в сверточном слое.

Первый слой СНС во всех случаях является сверточным, над объектом, проходящим через него, производятся следующие манипуляции. Заранее задается матрица весов, которая называется фильтром. На рис.7, например, он имеет размер

5x5. Далее фильтр направляется на часть входного изображения, которое также представляется в виде матрицы. Эта область, поражаемая фильтром, называется рецептивным полем. В итоге происходит поэлементное умножение двух матриц, а затем суммирование всех получившихся произведений. После этого действие фильтра сдвигается на заранее заданное число позиций (например, в самом простом случае - на одну), и вычисления повторяются. Данный процесс происходит до тех пор, пока не будет обработан весь исходный объект. В итоге получается новая матрица, состоящая из всех результатов действия фильтра. Она может быть проинтерпретирована следующим образом: в ее структуре содержатся характерные признаки изображения, например, прямые или кривые линии.

Результат работы первого слоя затем подается на следующий сверточный слой, для которого он уже будет являться входным объектом. Здесь процесс повторяется, но в итоге обнаруживаются более высокоуровневые свойства, представляющие собой композиции ранее найденных признаков. Количество сверточных слоев зависит от того, насколько глубокое понимание исходного объекта необходимо.

Нельзя не отметить еще один важный слой в СНС, который в основном является последним. Он называется полносвязным (fully connected). Механизм работы данного слоя заключается в соединении каждого нейрона предыдущего (например, сверточного) слоя с каждым нейроном, входящим в его состав. Описанная операция помогает сети обучиться запоминать, какие высокоуровневые признаки свойственны тому или иному классу.

2.2 Batch нормализация

При рассмотрении реализации DCGAN подхода нельзя обойти стороной такой метод, как batch нормализация, впервые предложенный в [12]. Его использование позволяет значительно сократить время обучения, что очень важно для сетей со сложной архитектурой. В целом процесс обучения СНС проходит с использованием так называемых батчей (batch). Каждый из них является

подвыборкой обучающего множества и поочередно подается на сеть. Для того, чтобы нормализовать батчи, и используется данная технология. Алгоритм действий заключается в следующем: первоначально вычисляется среднее и дисперсия для каждого конкретного батча (здесь m – число объектов в батче):

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2$$

Далее происходит непосредственная нормализация, в результате которой данные будут иметь среднее равным нулю и дисперсию равную единице.
Здесь ε – некоторое малое значение:

$$\dot{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

Заключительным этапом является применение двух новых переменных γ и β, которые добавляются к параметрам нейронной сети и настраиваются в процессе обучения.

$$y_i = \gamma\dot{x}_i + \beta$$

Значения y_i и являются результатом нормализации.

2.3 Архитектура дискриминативной DCGAN модели

Для реализации дискриминативной модели в данном параграфе будет использоваться теория сверточных сетей. В качестве обучающего множества вновь выберем набор рукописных цифр MNIST. Специфика архитектуры зависит от размера изображений в наборе данных. В данном случае он составляет 28x28x1, где третьим измерением является количество цветовых каналов. Здесь оно равно единице в связи с тем, что изображения являются монохромными. Если строить

модель для другого обучающего множества, например, с большим разрешением, то главным изменением будет варьирование количества сверточных слоев в сети.

В итоге было решено остановиться на следующей архитектуре (рис. 8). В ней использовались два подряд идущих сверточных слоя (на рис. 8 – серый цвет), с той лишь разницей, что первый сразу же сопровождался активационной функцией LReLu (на рис. 8 – зеленый), а во втором была дополнительно применена batch нормализация. Отсутствие ее в первом слое объясняется тем, что это позволит сети запомнить, каковым было входное распределение данных. Упомянутая здесь активационная функция LReLU (рис. 9) представляет из себя модификацию рассмотренной ранее функции ReLU, с той лишь разницей, что в этот раз отрицательные входы не превращаются в ноль, а принимают некоторые значения меньше нуля, которые по модулю растут намного медленнее, чем для положительных входов.

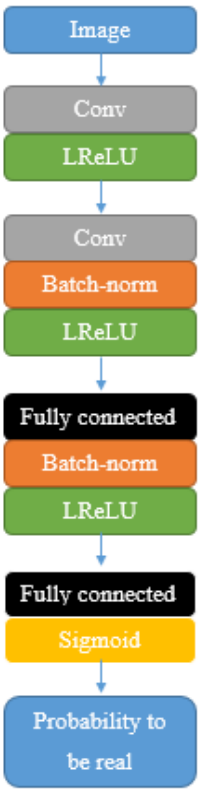

Рис.8: Архитектура дискриминативной модели.

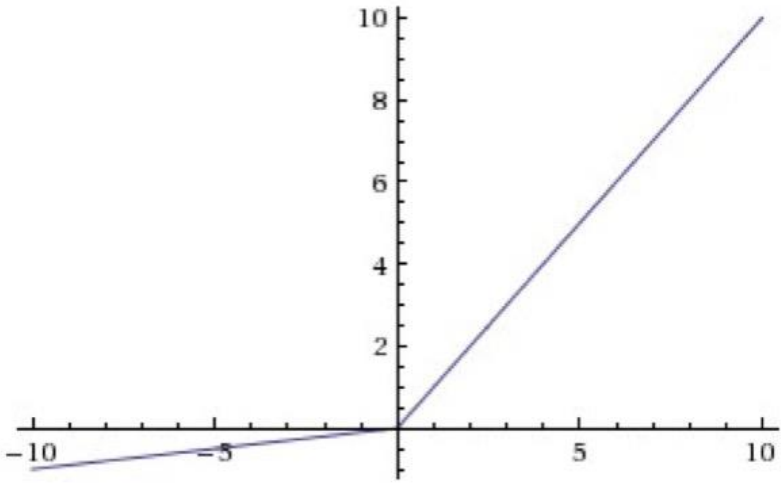

Рис. 9: Функция LReLU.

После идущих сверточных слоев следуют два полносвязных (на рис. 8 – черным цветом), которые заканчиваются сигмоидальной функцией активации (на рис. 8 – желтым цветом).

Представленная архитектура объясняется следующим образом: были выбраны фильтры и сдвиг этих фильтров для сверточных слоев таким образом, чтобы последующее рецептивное поле имело размер в два раза меньше, чем входная матрица. Данное соотношение было продемонстрировано в [13] для изображений размером 64x64, используя которое были достигнуты качественные результаты. Отсюда получается, что при изображениях размером 28x28 уменьшать их можно только два раза: сначала до 14x14, а затем до 7x7.

2.4 Архитектура порождающей DCGAN модели

Перед рассмотрением архитектуры для второй модели, а именно порождающей, будет произведена следующая поправка. Как описано в пункте 1.1 данной работы, на вход порождающей сети поступает шум, который по факту представляет из себя вектор произвольной размерности. Отсюда следует, что нет никакой необходимости расчленять его на признаки с помощью сверточных слоев, а требуется, наоборот, преобразовать в результирующее изображение. Для выполнения этой задачи будут использоваться так называемые разверточные (deconvolutional) слои, которые по факту являются обратными к сверточным. Принцип их действия заключается в том, что они дополняют вход нулями до матрицы необходимого размера. Выбранная архитектура представлена на рис. 10. Если внимательно рассмотреть и сравнить ее с устройством дискриминативной модели из прошлого параграфа, то можно заметить, что они между собой являются взаимно обратными, с точностью до замены сверточных слоев на разверточные, активационной функции LReLU на ReLU и сигмоидальной на гиперболический тангенс. В остальном, формально говоря, соотношения структур двух моделей можно представить в виде формулы:

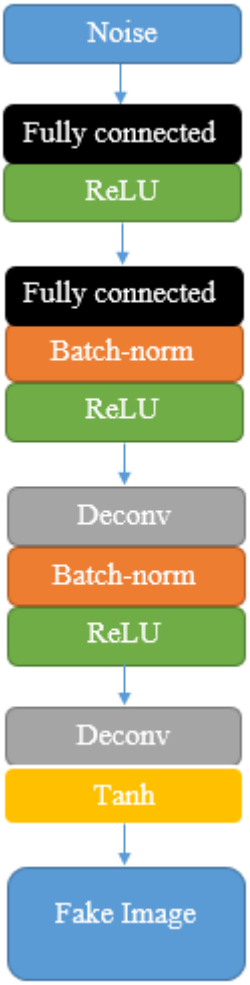

Рис. 10: Архитектура порождающей сети.

$$G = D^{-1},$$

где G – порождающая модель,

D – дискриминативная

Такое свойство носит не случайных характер, а вполне осознанный выбор. Все дело в том, что в процессе реализации данного подхода, были испробованы различные архитектуры сетей, в том числе такое соотношение как порождающая модель – персептрон, дискриминативная – сверточная. Или другое – оба «соперника» являются СНС, но с разным количеством слоев. Во всех результатах замечалась схожая тенденция: порождающая модель зацикливалась на одной или, в лучшем случае, на нескольких цифрах и старалась повторить только их, а не охватить всевозможные вариации. Примеры таких результатов можно увидеть на рис. 11.

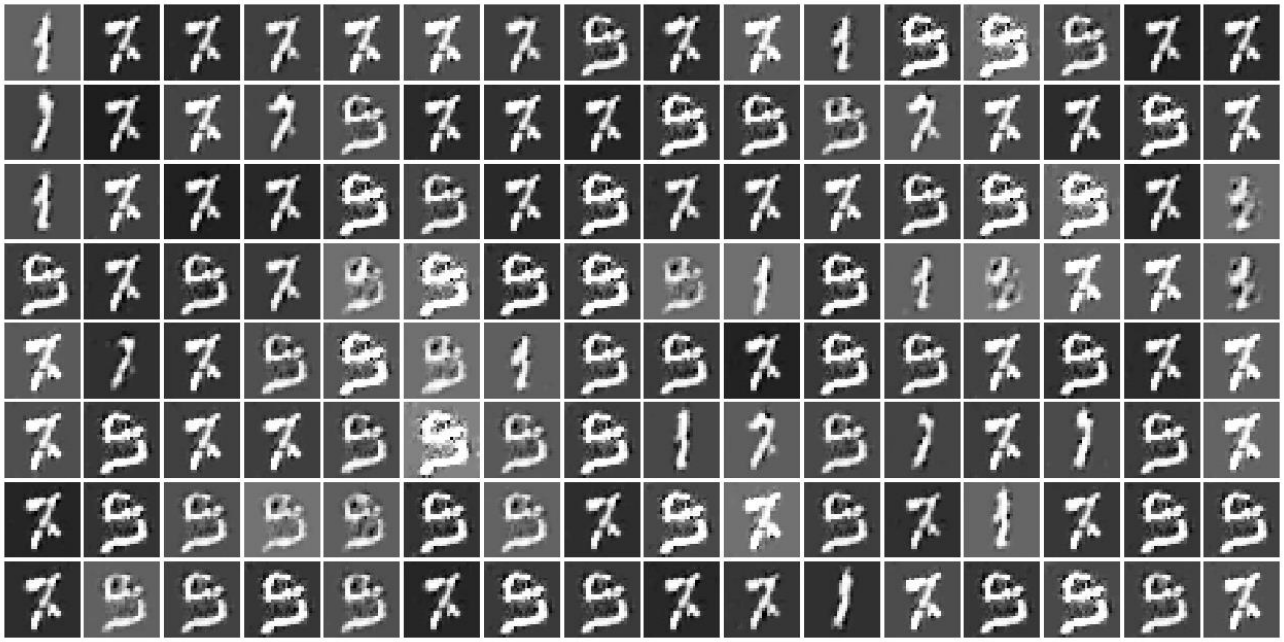

Рис. 11: Пример результатов, когда GAN модели имели совершенно разные структуры.

Впрочем, нельзя с уверенностью утверждать, что наличие одинаковых архитектур носит необходимый характер для успешности GAN подхода, но как минимум это ускоряет процесс обучения и улучшает качество результатов на порядок.

2.5 Результаты DCGAN подхода

Итогом взаимодействия двух моделей, рассмотренных в пунктах 2.3 и 2.4, являются рукописные цифры, представленных на рис.12. Анализируя полученные изображения, можно констатировать, что подавляющее большинство являются хорошо идентифицированными, качество написания которых в общем не должно вызывать вопросов. В сравнении с результатами работы персептронов необходимо заметить, что качество изображений значительно изменилось в лучшую сторону. Из всего этого можно сделать вывод, что DCGAN модель обучена настолько, что создаваемые ей цифры достаточно схожи с реальными. Достигнув этого, появляется возможность перехода к основной задаче – решению проблемы увеличения разрешения изображения.

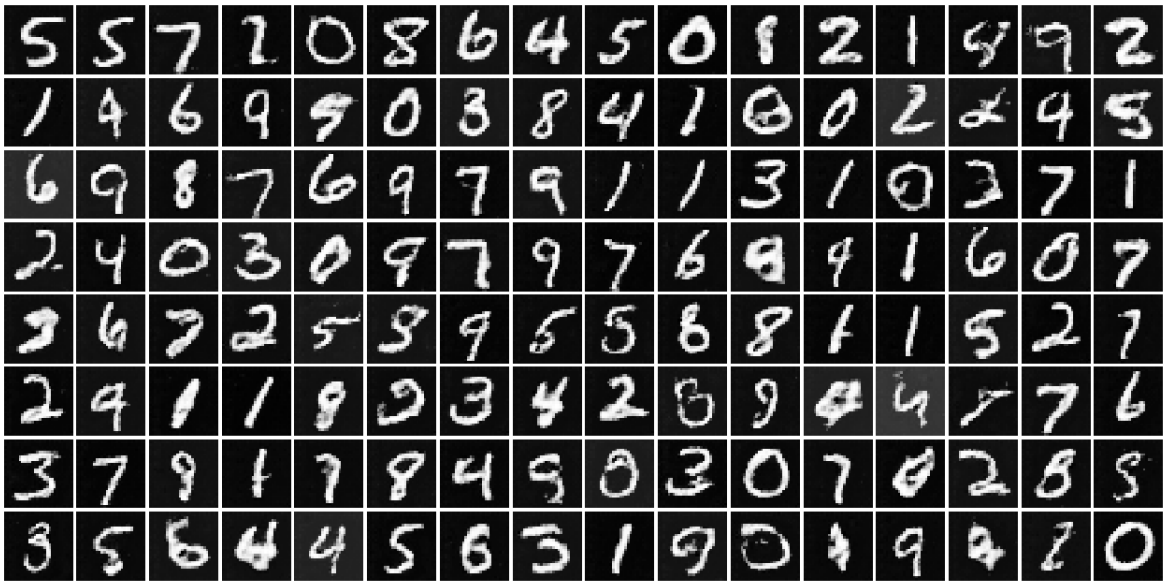

Рис. 12: Результаты DCGAN.

Также в приложении А приведены графики изменения функций ошибок обеих моделей во время обучения. Анализируя их поведение, можно сделать вывод, что для успешности DCGAN важно, чтобы значения ни одной из функций

не стремились к нулю. Данное свойство в полной мере характеризует основной принцип рассматриваемого подхода – успех будет достигнут только в том случае, если обе сети сохраняют баланс.

Глава 3. Повышение разрешения изображения

3.1 Алгоритм Inpainting с применением GAN подхода

Впервые алгоритм дорисовки (Inpainting), включающий в себя применение GAN подхода, был описан в [6]. В целом он представляет из себя некоторую надстройку над уже обученной создавать свои собственные изображения модели, другими словами является одним из возможных применений данного подхода.

Дорисовка – это процесс восстановления, реконструкции изображения, имеющего некоторые дефекты. Например, зашумленность или полная потеря одной из частей фотографии. Алгоритм, который впоследствии будет использован для задачи повышения разрешения, авторами статьи [6] создавался именно для возможности реставрировать поврежденные изображения, а не для увеличения разрешения фотографий. В связи с этим, сначала опишем данный алгоритм в контексте его основной задачи.

Пусть имеется исходное изображение $\mathbf{Y}$, которое будет подвержено некоторому дефекту, и бинарная маска $\mathbf{M}$, которая реализует данное повреждение. Фактически, $\mathbf{M}$ – матрица, при умножении на которую качественное изображение превращается в испорченное. Также в распоряжении имеется уже обученная DCGAN модель из главы 2. Предположим, что найден такой шум $\hat{\mathbf{z}}$, что порождающая модель $G(\hat{\mathbf{z}})$ воссоздает исходное изображение без каких-либо повреждений. Тогда реконструированная фотография примет вид:

$$\mathbf{X}_{\mathbf{reconstructed}} = \mathbf{M} \cdot \mathbf{Y} + (1 - \mathbf{M}) \cdot G(\hat{\mathbf{z}})$$

Теперь достаточно будет найти необходимый шум $\hat{\mathbf{z}}$. Для этого потребуется ввести два новых понятия: контекстуальная (contextual) и перцептуальная (perceptual) информация. Контекстуальная информация – это сведения о пикселях изображения, которые расположены вокруг дефекта. Она является крайне важной для того, чтобы качественные пиксели исходного изображения максимально совпадали с пикселями, созданными порождающей моделью. Другими словами,

эта информация позволяет запомнить неповрежденные части фотографии. Чтобы добиться данного эффекта, порождающая сеть будет штрафоваться, если созданное ею изображение будет отличаться от исходного.

$$L_{contextual}(z) = ||\mathbf{M} \cdot G(z) - \mathbf{M} \cdot \mathbf{Y}||_1,$$
$$где\ ||x||_1 = \sum |x_i|$$

В идеальном случае контекстуальная информация должна равняться нулю, тогда все качественные пиксели совпадут друг с другом.

Вторая введенная величина – это перцептуальная информация. Она показывает насколько достроенная часть изображения выглядит правдоподобно. Именно эта величина и позволяет достойным образом дорисовывать зашумленную часть изображения. Здесь будет использована уже обученная дискриминативная модель, которая по факту и определяет, выглядит объект реально или нет:

$$L_{perceptual}(z) = \log(1 - D(G(z)))$$

Теперь все готово к тому, чтобы найти необходимый шум $\hat{\mathbf{z}}$, с помощью которого порождающая сеть и создаст достроенное изображение. Взглянув еще раз на введенные прежде величины-информации, нетрудно заметить, что для достижения максимального эффекта от дорисовки, необходимо минимизировать как концептуальную, так и перцептуальную информацию. Отсюда возникает следующая задача оптимизации:

$$L(z) = L_{contextual}(z) + \lambda L_{perceptual}(z)$$
$$\hat{\mathrm{z}} = arg\min_{z} L(z),$$
где λ – весовой коэффициент.

После нахождения $\hat{\mathbf{z}}$ потребуется воспользоваться уже приведенной формулой и найти реконструированное изображение:

$$x_{reconstructed} = \boldsymbol{M} \cdot \boldsymbol{Y} + (1 - \boldsymbol{M}) \cdot G(\hat{\mathrm{z}})$$

Результаты работы данного алгоритма, которые были продемонстрированы в [6], можно наблюдать на рис. 13. Как видно, при достаточно большом наборе тренировочных данных (использовалось приблизительно 200 000 изображений) достигается весьма неплохой успех. Для обучения использовался пакет данных CelebA.

Рис.13: Результаты работы алгоритма Inpainting, представленные в [6]. Первая колонка – исходное изображение, вторая – зашумленное на 80% изображение, третья – применение алгоритма к второй колонке, четвертая – полное зашумление центральной части изображения, пятая – применение алгоритма к четвертой колонке.

3.2 Применение Inpainting для задачи повышения разрешения

Теперь адаптируем описанный алгоритм для использования в основной задаче, рассматриваемой в работе, т.е. задаче повышения разрешения изображений. Фактически, на вход алгоритм Inpainting получает некоторое зашумленное изображение. Шум может располагаться как угодно, даже случайным образом. Следовательно, ничто не мешает расположить его упорядоченно, а именно: расставить пиксели изображения низкого разрешения на регулярную решетку с фиксированным шагом, который определяется в зависимости от того, во сколько раз изображение должно быть увеличено. Все пиксели между узлами заполнить нулями. Эти нули и будут определять структуру маски **M**, которая была описана ранее. Иллюстрацию данного процесса можно наблюдать на рис. 14.

Далее, чтобы использовать представленный алгоритм, достаточно изображение, разрешение которого необходимо повысить, подвергнуть данной процедуре. После нее увеличенная фотография будет полностью удовлетворять требованиям алгоритма.

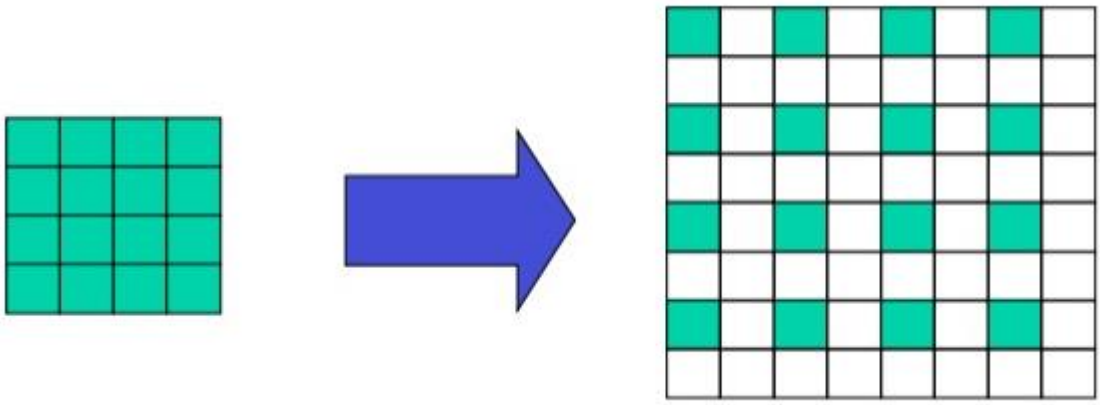

Рис. 14: Увеличение изображения в два раза с помощью размещения его на регулярной решетке и заполнения оставшегося места нулями.

3.3 Алгоритм SRGAN

Другим подходом решения задачи повышения разрешения изображения является специализированный алгоритм, основанный на использовании GAN подхода и называющийся SRGAN (Super-Resolution Generative Adversarial Networks). Он был разработан и представлен в [7]. Концептуально он мало отличается от уже описанных конкурирующих нейронных сетей, но все же имеет некоторые особенности.

Во-первых, на вход порождающей сети подается не случайный шум, как в классическом GAN подходе, а изображения низкого разрешения, которое необходимо будет увеличить. Выходом, соответственно, является увеличенное изображение. Задача дискриминативной модели остается прежней – ей необходимо определять, какая фотография является оригиналом входного изображения, а какая – продуктом работы порождающей модели.

Во-вторых, претерпело некоторое изменение архитектура сетей, а именно порождающей модели. В нее перед разверточными слоями добавляется пара сверточных слоев, отвечающих за то, чтобы первоначально исходное изображение декомпозировать на признаки, а затем уже благодаря последующим преобразованиям превратить полученные свойства в изображение большего разрешения. При этом архитектура дискриминативной сети остается неизменной.

3.4 Метрики качества оценки результатов

Для проведения сравнительного анализа работы двух алгоритмов помимо визуального восприятия и понимания, какое изображение выглядит качественней, необходимы формальные критерии. В данном параграфе будут рассмотрены некоторые показатели, которые было бы уместно использовать для оценки результатов поставленной задачи.

Первой рассматриваемой метрикой будет MSE (Mean Squared Error) – среднеквадратичная ошибка:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2,$$

где m, n – размер сравниваемых изображений I и K.

Главное преимущество данной метрики – простота реализации. Но сразу же возникает и главная проблема – далеко не всегда большая разница между значениями пикселей является показателем того, что фотографии сильно отличаются друг от друга визуально. Данный момент хорошо иллюстрирует следующий пример: на рис. 15a расположены два изображения, на которых помещен одинаковый объект, но с разной степенью контрастности пикселей. На рис. 15b можно наблюдать две фотографии, значительно отличающиеся друг от друга. Проблема MSE заключается в том, что во втором случае его показатель меньше, чем в первом, что, естественно, не сопоставляется с человеческим визуальным восприятием и представлением о схожести изображений.

Рис. 15: Пример проблемы использования MSE.

Второй метрикой предлагается использовать PSNR (peak-signal-to-noise ratio). Она отражает соотношение между максимум возможного значения пикселей изображения и мощностью шума, воздействующего на данное изображение. PSNR вычисляется с помощью описанной ранее среднеквадратической ошибки:

$$PSNR = 20log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right),$$

где MAX_I – максимально возможное значение пикселей на изображении I.

Данный показатель часто используется для сравнения исходного и искаженного изображения (что в принципе и необходимо для поставленной задачи), единицей измерений являются децибелы (dB).

Третьей и последней предлагаемой метрикой является SSIM. Ее отличительной чертой можно назвать тот факт, что она учитывает структурные свойства изображения, и несколько больше соответствует особенностям человеческого восприятия. Данный показатель формально представляется в следующем виде:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

где

- μ_x – *среднее изображения x*
- μ_y – *среднее изображения y*
- σ_x^2 – *дисперсия изображения x*
- σ_y^2 – *дисперсия изображения y*
- σ_{xy} – *ковариация изображений x и y*
- $c1, c2$ – *некоторые константы*

Если сравнивать эту метрику с MSE, можно заметить, что на рассмотренном примере на рис. 15 SSIM изображений a) равен 0.78, а изображений b) – 0.69, что представляется более адекватным в сравнении с результатом по MSE.

Выводы

После применения двух предложенных алгоритмов к набору рукописных цифр MNIST были получены следующие результаты (рис.16). Заметим, что в данном случае увеличение происходило в два раза.

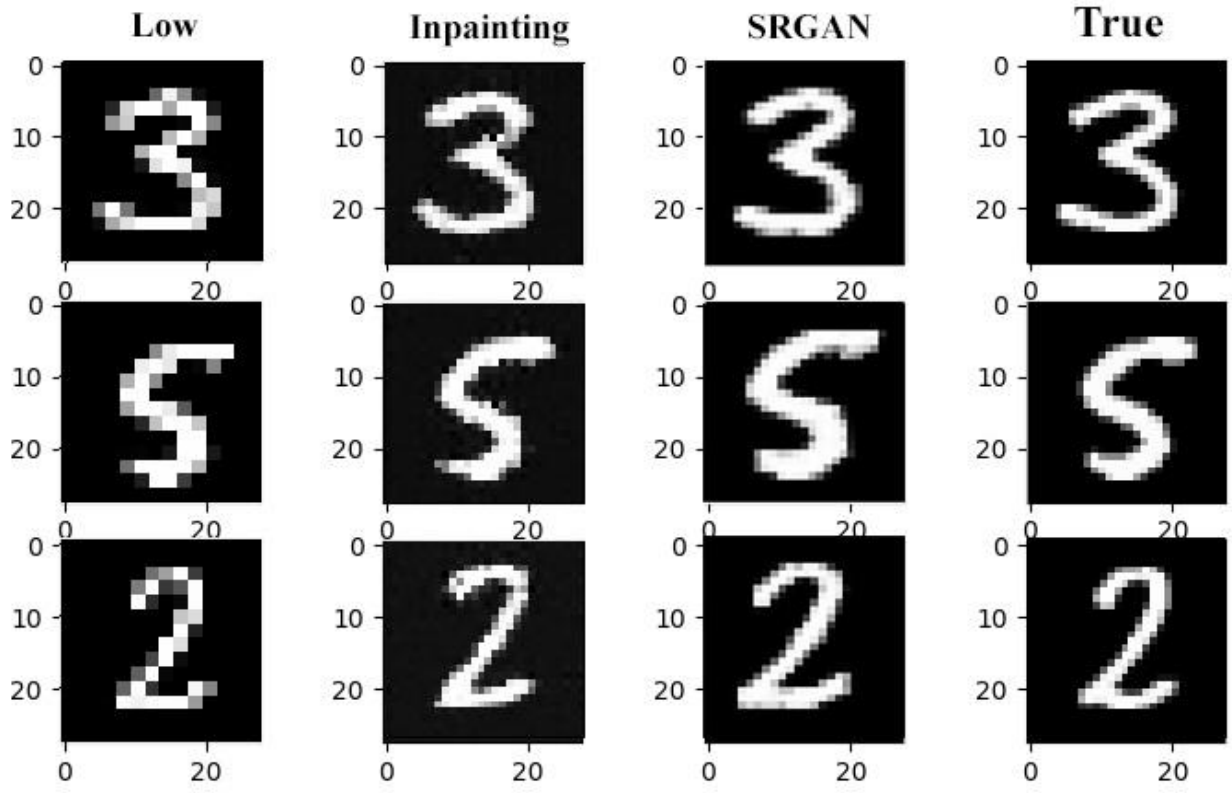

Рис. 16: Результаты повышения разрешения в два раза: первая колонка – исходное изображение низкого качества, вторая колонка – применение алгоритма Inpainting, третья колонка – применение алгоритма SRGAN, четвертая колонка – исходное изображение высокого качества.

Из анализа получившихся данных можно сделать вывод, что оба алгоритма справились со своей задачей, в том смысле, что разрешения изображений действительно были увеличены. При этом результаты не потеряли своей «реальности» и остались идентифицируемыми, т.е. объект, который находился на фотографии низкого качества, остался тем же самым объектом после применения обоих алгоритмов. Переходя к сравнению второй и третьей колонки на рис. 16, отчетливо видно, что специализированный алгоритм SRGAN справился лучше, чем Inpainting, изображения получились более резкими, пиксели различимы

исключительно на границе, где белый цвет переходит в черный.

Цифра (из рис. 99)	*Inpainting*	*SRGAN*
 (3)	MSE: 1084.75 SSIM: 0.78 PSNR: 17.7 dB	MSE: 564.34 SSIM: 0.89 PSNR: 20.6 dB
 (5)	MSE: 1456.19 SSIM: 0.81 PSNR: 16.5 dB	MSE: 721.05 SSIM: 0.92 PSNR: 19.5 dB
 (6)	MSE: 660.58 SSIM: 0.89 PSNR: 19.9 dB	MSE: 707.88 SSIM: 0.92 PSNR: 19.6 dB

Рис.17: Сравнения показателей MSE и SSIM для работы алгоритмов Inpainting и SRGAN.

Из анализа таблицы, представленной на рис. 17, можно также сделать вывод, что SRGAN повышает разрешение изображений качественнее. Необходимо отметить, что только на цифре 6 MSE и PSNR у алгоритма Inpainting оказался выше. При этом во всех остальных случаях все показатели SRGAN превосходят своего конкурента. С более подробной выкладкой результатов можно ознакомиться в приложении Б. В нем есть одно интересное изображение – цифры 4 – которая нарисована несколько не совсем привычным образом. Можно констатировать, что с ней Inpainting совершенно не справился (SSIM = 0.5), хотя при этом SRGAN выполнил свою работу качественно (SSIM = 0.89).

Несмотря на все вышеуказанные преимущества SRGAN, алгоритм Inpainting также продемонстрировал достойные результаты, и существенно не уступает (особенно по метрике SSIM) от своего более продвинутого «коллеги». При этом нужно отметить одно бесспорное достоинство дорисовки, заключающееся в том, что Inpainting является надстройкой над обученной на изображениях какого-то

фиксированного разрешения (назовем его **NxN**) GAN модели. Отсюда следует, что абсолютно неважно, фотографии какого размера будут увеличиваться. Разрешение может повышаться в два раза, как в приведенных результатах. В них изображения, которые увеличивались, имели размер $\frac{N}{2}\mathbf{x}\frac{N}{2}$. При этом ни что не мешает воспользоваться, например, изображениями с разрешением $\frac{N}{4}\mathbf{x}\frac{N}{4}$ или $\frac{N}{8}\mathbf{x}\frac{N}{8}$. Единственные изменения будут заключаться в том, что при подготовке объекта к применению алгоритма Inpainting больше пикселей будут заполнены нулями (пункт 3.2). Модель SRGAN же, напротив, потребует полного переобучения сетей, а возможно даже и изменения архитектуры порождающей модели, т.к. она изначально зависит от того числа, характеризующего во сколько раз изображение будет увеличиваться. Процедура обучения сети занимает продолжительное время, что, конечно же, является отрицательным фактором.

Заключение

В данной работе была рассмотрена проблема повышения разрешения изображения и возможности ее решения с помощью относительно нового GAN подхода. Главной целью являлось сравнение двух алгоритмов на базе данного подхода: алгоритма Inpainting и алгоритма SRGAN. Первый из них создан для более общего типа задач дорисовки и напрямую к данной конкретной проблеме до этого не применялся. Второй же, напротив, является специально разработанным алгоритмом для задачи повышения разрешения. Основной вопрос заключался в том, справится ли алгоритм Inpainting с поставленной перед ним задачей, т.е. будут ли увеличенные изображения являться качественными, и если справится, то насколько сильно он будет уступать алгоритму SRGAN.

В процессе анализа данной проблемы первоначально был успешно реализован сам GAN подход, показаны отличия в результатах, которые возникают, если использовать различные нейронные сети в качестве конкурирующих моделей. В частности, были приведены примеры использования многослойных персептронов и сверточных сетей. На последних и было решено остановиться, т.к. они смогли продемонстрировать, что обладают возможностью создавать качественные, мало отличимые от реальных, изображения.

Затем к уже обученной GAN модели, состоящей из сверточных нейронных сетей, был применен алгоритм Inpainting, благодаря которому данная модель получила возможность увеличивать разрешение фотографий. После этого была создана совершенно отдельная нейросетевая архитектура для алгоритма SRGAN.

В итоге сравнение результатов работ двух данных алгоритмов показало, что по субъективной визуальной оценке и по формальным аналитическим метрикам качества алгоритм SRGAN справился с задачей лучше. При этом нельзя утверждать, что алгоритм Inpainting сильно отстал: значения показателя SSIM, использованного для оценки полученных изображений, говорят о том, что разница в качестве между алгоритмами является весьма незначительной. К тому же он обладает неоспоримым преимуществом перед SRGAN в том, что при изменении

мультипликатора нет необходимости переучивать GAN модель заново.

Таким образом, из вышесказанного можно сделать вывод, что алгоритм Inpainting имеет смысл применять в следующих ситуациях:

- имеется уже обученная GAN модель для каких-либо других целей, при этом поставлена задача увеличения фотографий (особенно в короткие сроки)
- заранее неизвестно, во сколько раз потребуется увеличивать фотографию, или, напротив, есть уверенность, что будет необходимо производить повышение разрешения с разным мультипликатором

Во всех представленных случаях применение алгоритма Inpainting полностью оправдано, т.к. его реализация не требует перенастройки GAN модели. Если же имеется достаточно времени для полного переобучения нейронных сетей и требуется добиться наилучшего качества, то, бесспорно, выбор специализированного алгоритма SRGAN не должен вызывать вопросов.

Список литературы

1. Neural Networks and Deep Learning, http://neuralnetworksanddeeplearning.com/
2. Ian J.Goodfellow, Jean Pouget-Abadie Generative Adversarial Nets // Departement d'informatique et de recherche operationnelle Universite de Montreal, 10 June 2014
3. Ian J.Goodfellow NIPS 2016 Tutorial: Generative Adversarial Networks, 3 Apr 2017
4. Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, Honglak Lee Generative Adversarial Text to Image Synthesis // University of Michigan, 5 Jun 2016
5. Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, Serge Belongie Stacked Generative Adversarial Networks // Department of Computer Science, Cornell University, 12 Apr 2017
6. Raymond Yeh, Chen Chen, Teck Yian Lim, Mark Hasegawa-Johnson, Minh N. Do Semantic Image Inpainting with Perceptual and Contextual Losses // Dept. of Electrical and Computer Engineering University of Illinois at Urbana-Champaign 14 Nov 2016
7. Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network // Twitter, 13 Apr 2017
8. Ryan Dahl, Mohammad Norouzi, Jonathon Shlens Pixel Recursive Super Resolution // Google Brain, 22 Mar 2017
9. Mehdi S. M. Sajjadi, Bernhard Scholkopf, Michael Hirsch EnhanceNet: Single Image Super-Resolution through Automated Texture Synthesis // Max-Planck-Institute for Intelligent Systems Spemanstr, 23 Dec 2016
10. Silvano Galliani, Charis Lanaras, Dimitrios Marmanis, Emmanuel Baltsavias, Konrad Schindler Learned Spectral Super-Resolution // Photogrammetry and Remote Sensing, ETH Zurich, Switzerland, 28 Mar 2017

11. Хайкин С. Нейронные сети. Полный курс. Второе издание. Москва: Изд. дом Вильямс, 2006. 225 с.
12. Sergey Ioffe, Christian Szegedy Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // Google Inc., 2 Mar 2015
13. Alec Radford, Luke Metz, Soumith Chintala Unsupervised representationg learning with deep convolutional generative adversarial networks, 7 Jan 2016

Приложение А

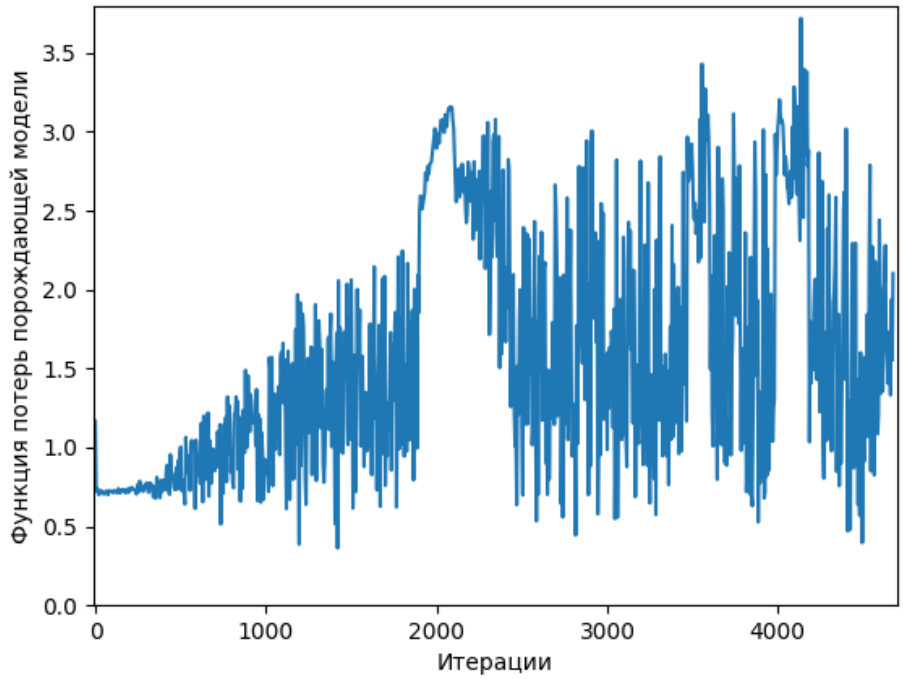

Рис.1: Поведение функции потерь порождающей модели во время обучения.

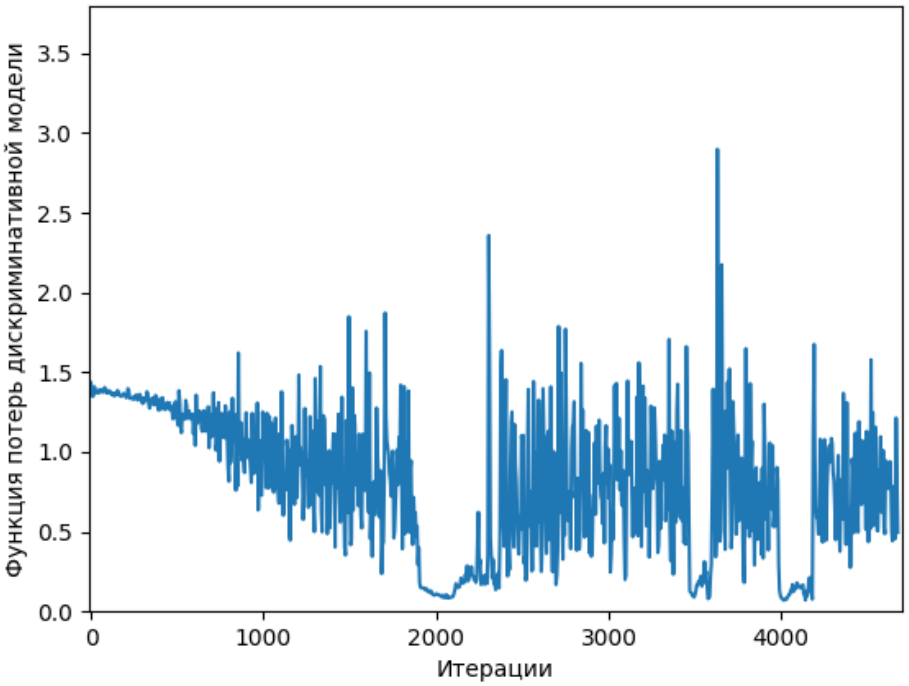

Рис.2: Поведение функции потерь дискриминативной модели во время обучения.

Приложение Б

Цифра	Inpainting	SRGAN
 **(1)**	MSE: 338.43 SSIM: 0.92 PSNR: 22.8 dB	MSE: 174.86 SSIM: 0.96 PSNR: 25.7 dB
 **(8)**	MSE: 927.29 SSIM: 0.87 PSNR: 18.4 dB	MSE: 596.91 SSIM: 0.92 PSNR: 20.4 dB
 **(9)**	MSE: 594.22 SSIM: 0.87 PSNR: 20.4 dB	MSE: 525.19 SSIM: 0.93 PSNR: 20.9 dB
 **(4)**	MSE: 3810.98 SSIM: 0.50 PSNR: 12.3 dB	MSE: 925.45 SSIM: 0.89 PSNR: 18.5 dB
 **(0)**	MSE: 1207.53 SSIM: 0.84 PSNR: 17.3 dB	MSE: 611.97 SSIM: 0.93 PSNR: 20.3 dB
 **(2)**	MSE: 509.79 SSIM: 0.88 PSNR: 21 dB	MSE: 549.93 SSIM: 0.89 PSNR: 20.7 dB
 **(5)**	MSE: 1367.83 SSIM: 0.83 PSNR: 16.8 dB	MSE: 740.97 SSIM: 0.91 PSNR: 18.4 dB
 **(7)**	MSE: 997.89 SSIM: 0.80 PSNR: 18.1 dB	MSE: 689.48 SSIM: 0.87 PSNR: 19.7 dB

Рис.1: Сравнение метрик качества для алгоритмов Inpainting и SRGAN.

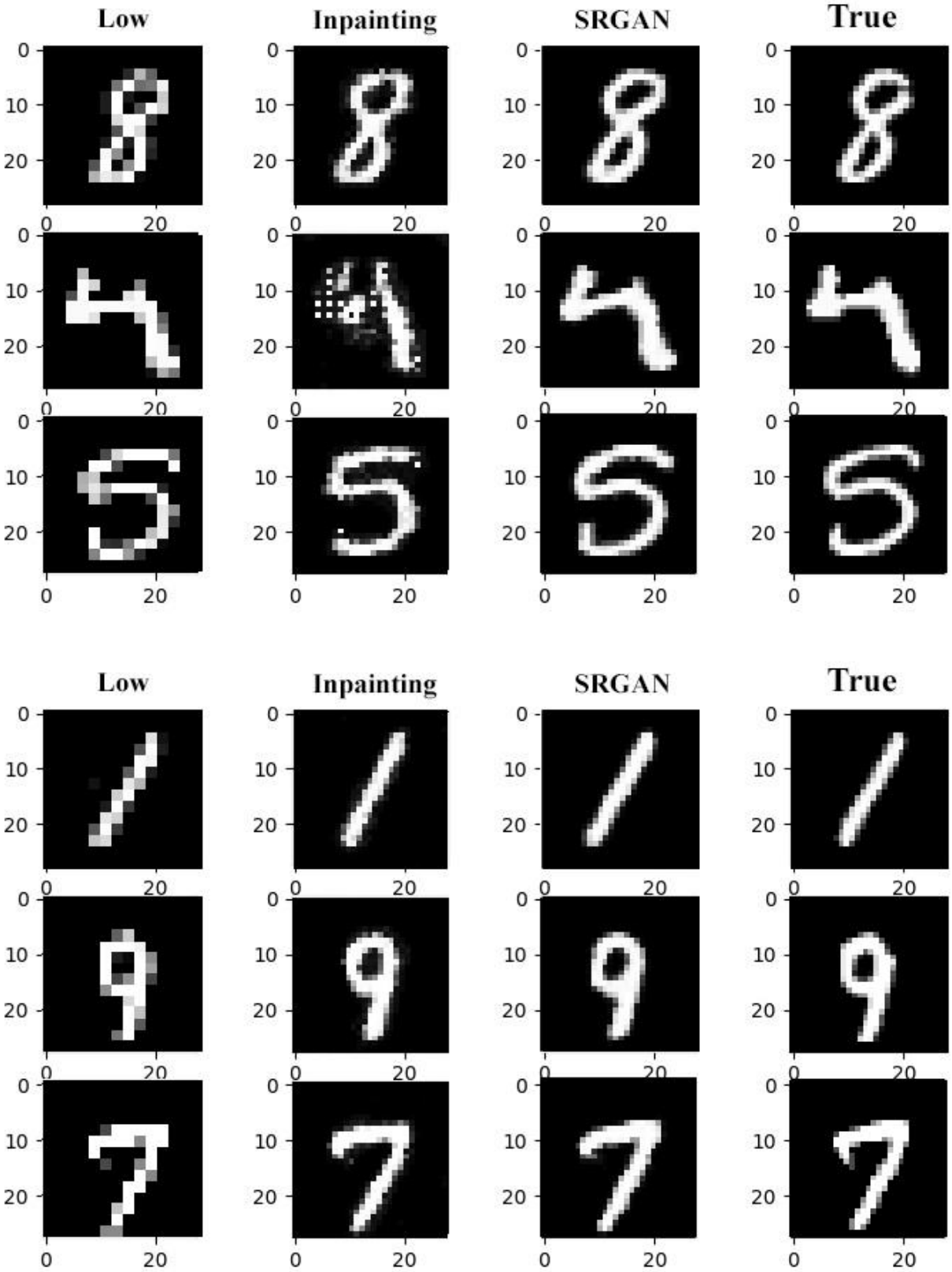

Рис. 2: Результаты повышения разрешения в два раза: первая колонка – исходное изображение низкого качества, вторая колонка – применение алгоритма Inpainting*, третья колонка – применение алгоритма SRGAN, четвертая колонка – исходное изображение высокого качества.*